


Redictado Taller de Programación 2021

CLASE 1

Ordenación de vectores

A silver laptop sits on a light-colored wooden desk. The laptop screen is open and displays a simple Pascal program. The background is a plain, light-colored wall.

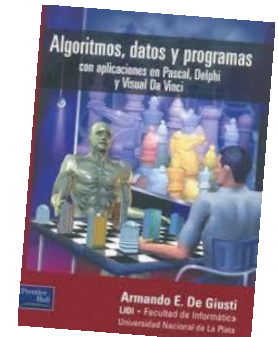
```
Program HolaMundo;  
Begin  
  writeln('Hola mundo');  
end.
```

TEMAS DE LA CLASE

1. Concepto de Ordenación
2. Método de Ordenación por Inserción
 - *Pseudocódigo*
 - *Análisis de casos*
 - *Análisis teórico*

Bibliografía:

Cap. 8. Algoritmos, Datos y Programas. *Armando E. De Giusti*.



1. CONCEPTO DE ORDENACIÓN.

Algoritmo de Ordenación

Proceso por el cual, un grupo de elementos puede ser ordenado.

¿Por qué es importante una operación de ordenación en arreglos?

Analicemos distintas situaciones problemáticas donde ésta operación es necesaria...

1. CONCEPTO DE ORDENACIÓN.

MÉTODOS DE ORDENACIÓN CLÁSICOS

Selección

Intercambio

Inserción

2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

Se parte de una secuencia de dos ítems y se ordena.

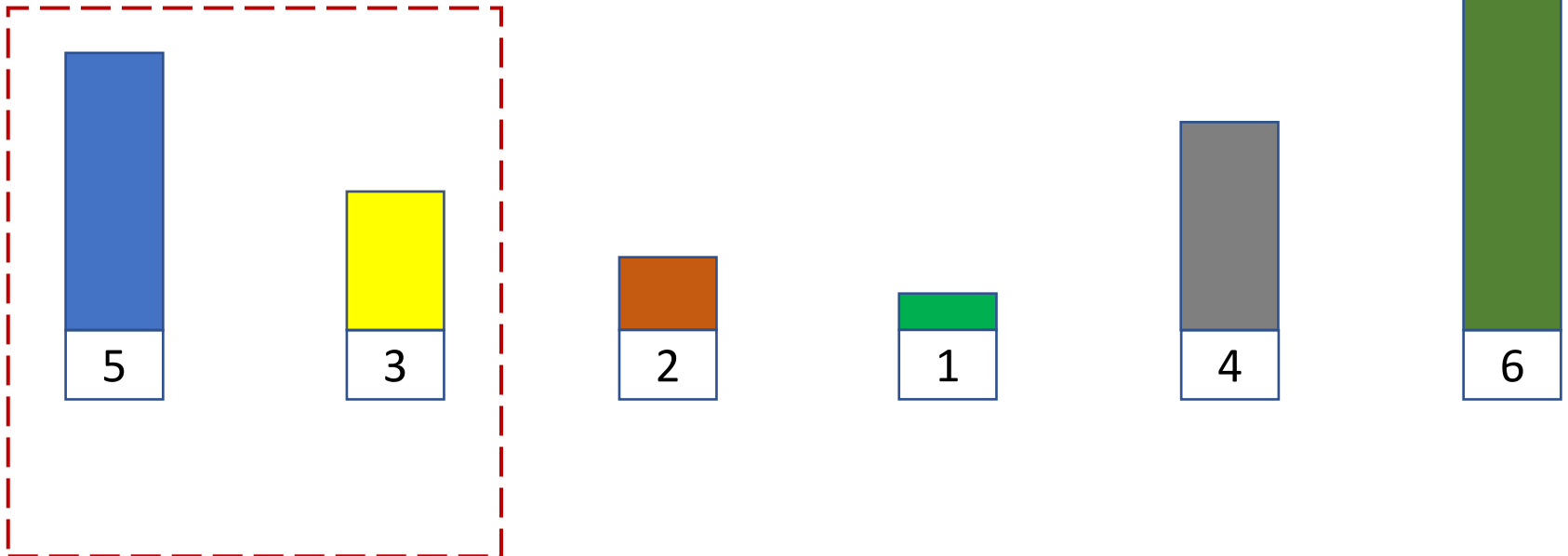
En cada pasada se “*agrega*” un ítem y se inserta en la posición correspondiente en el arreglo ordenado.

Veamos el ejemplo...

2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

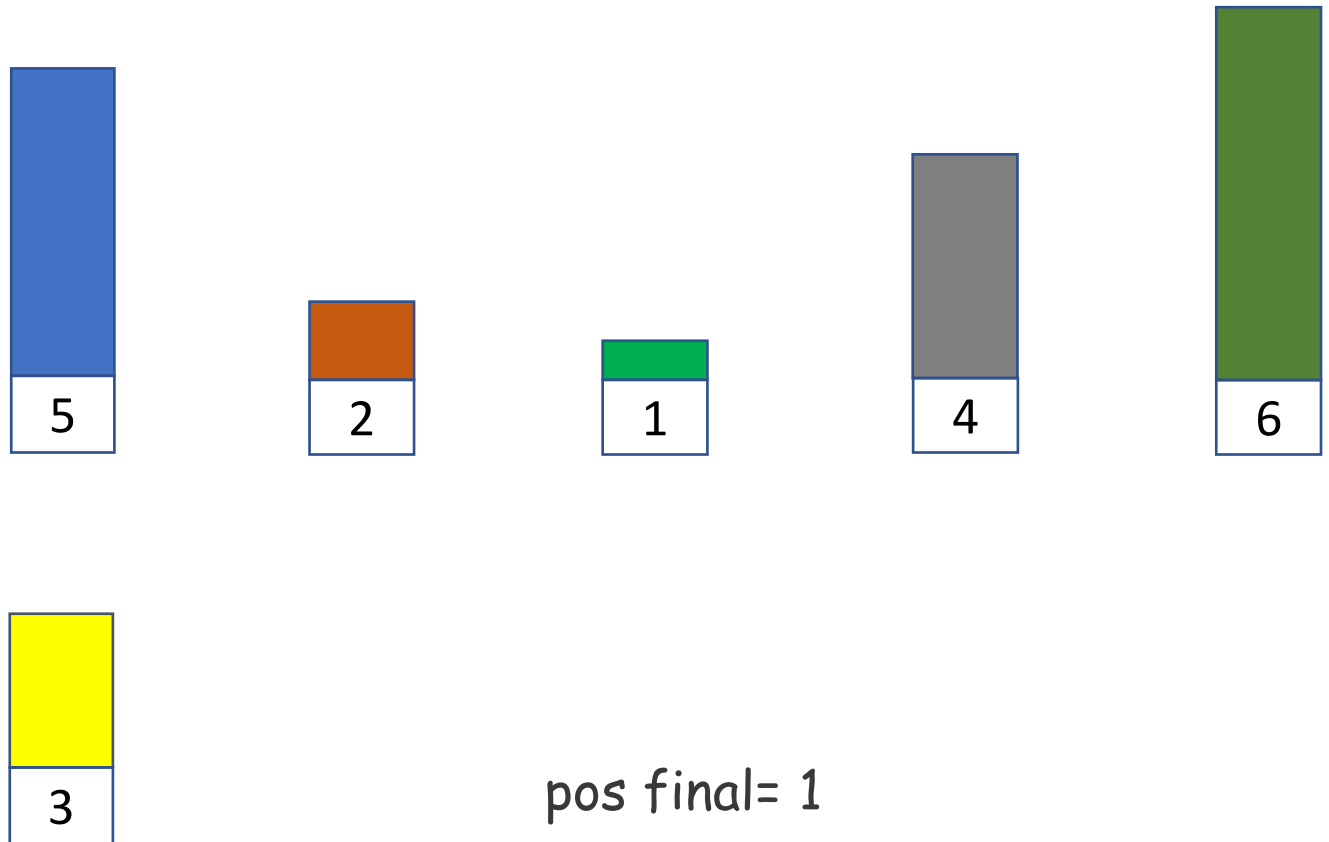
$i=2$ elemento a ordenar = 3



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

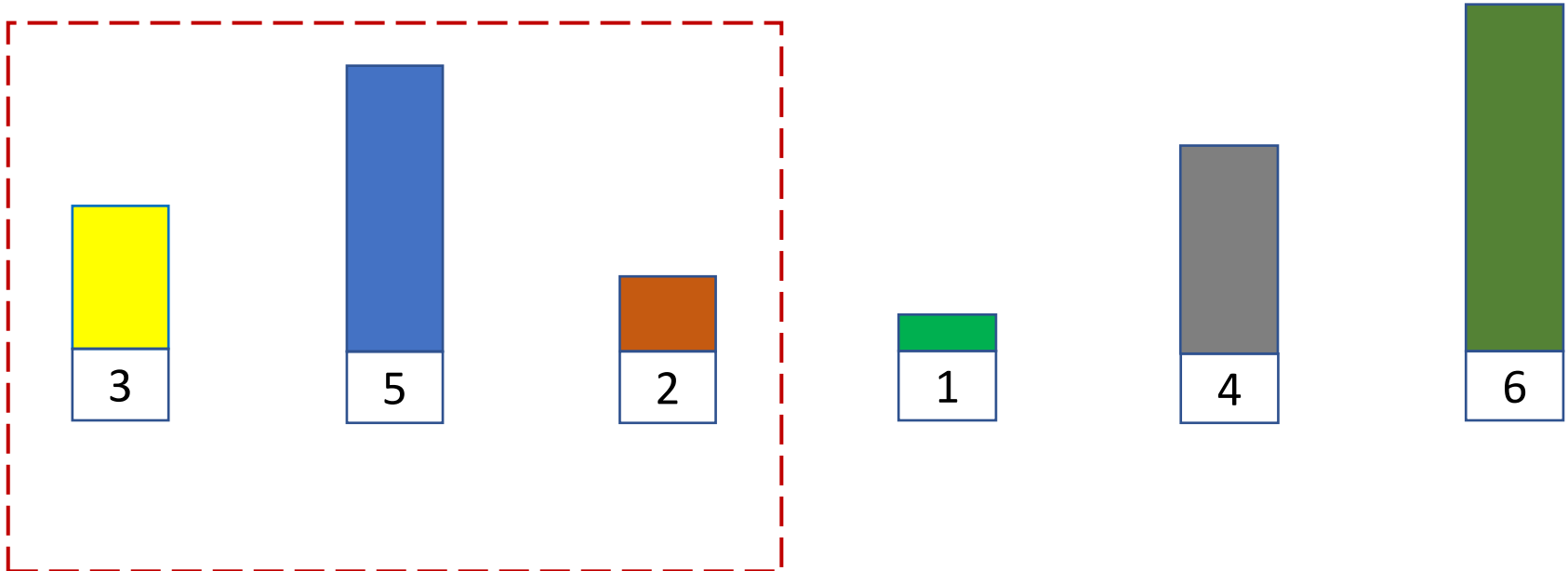
$i=2$ elemento a ordenar = 3



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

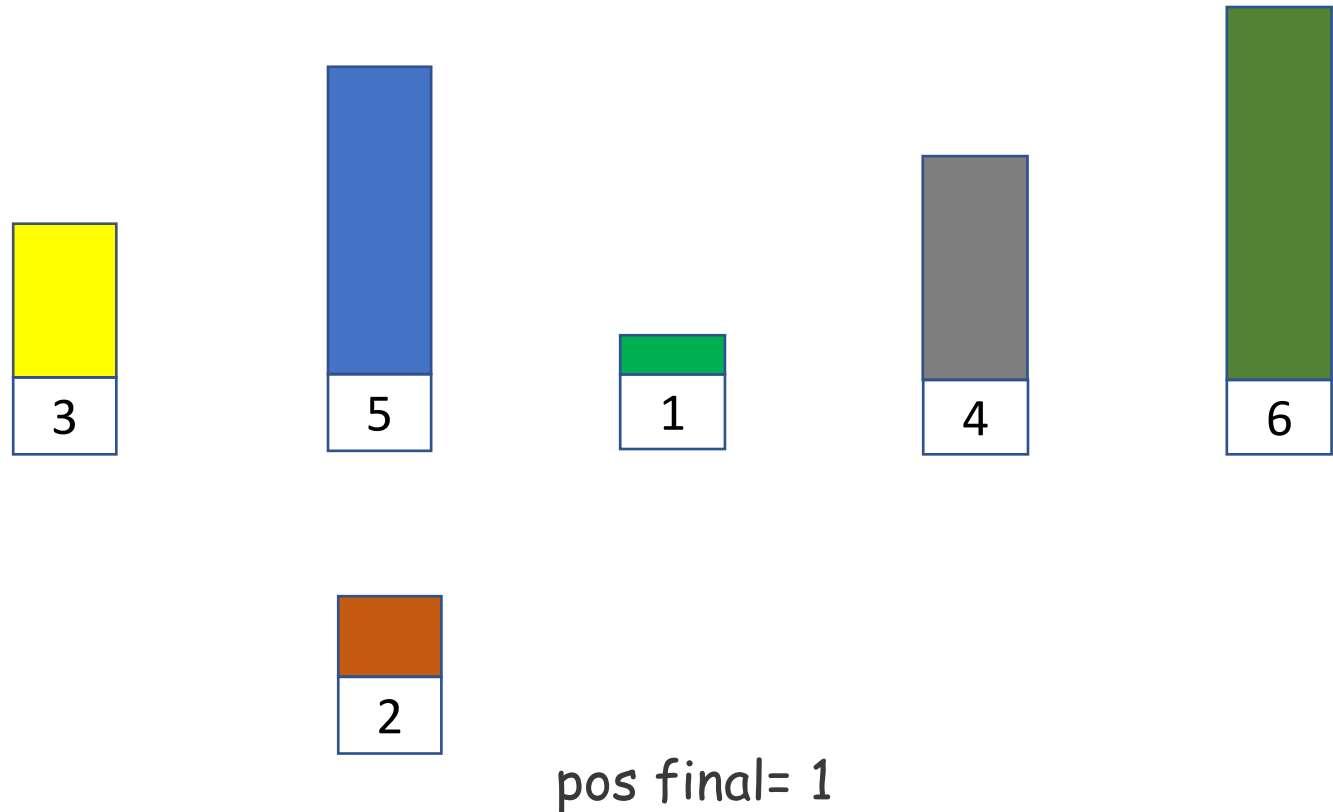
$i=3$ elemento a ordenar = 2



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

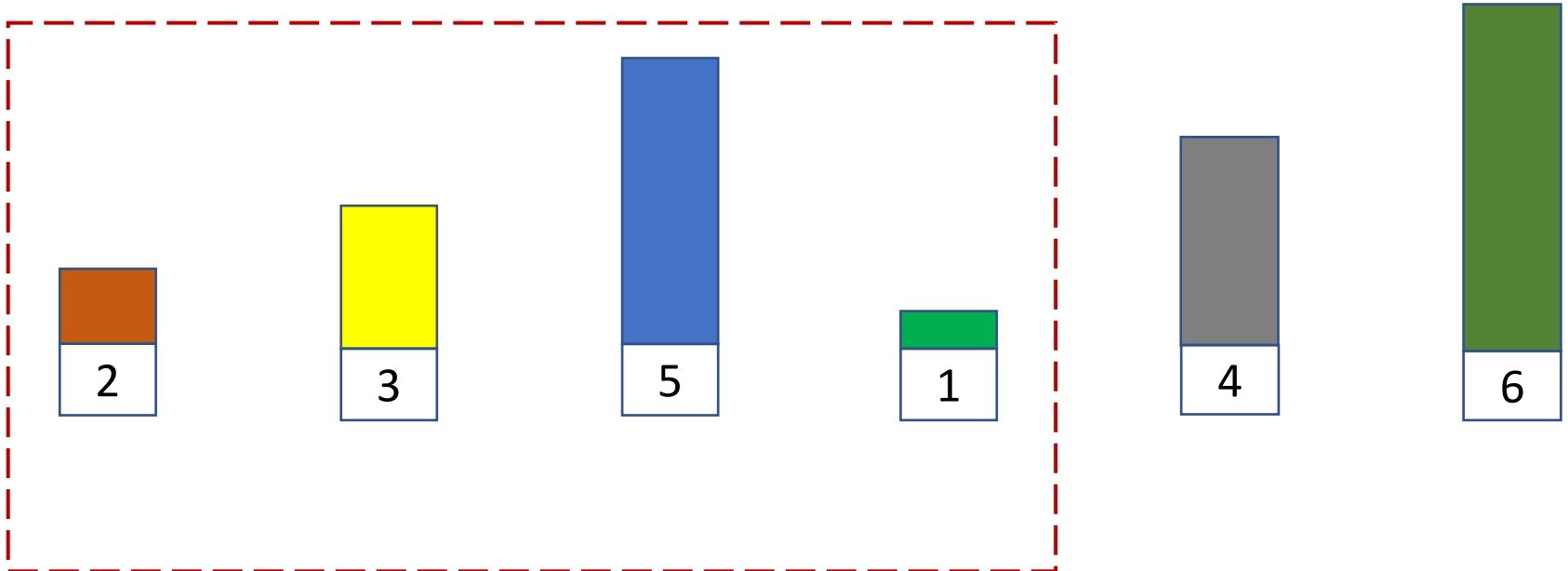
$i=3$ elemento a ordenar = 2



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

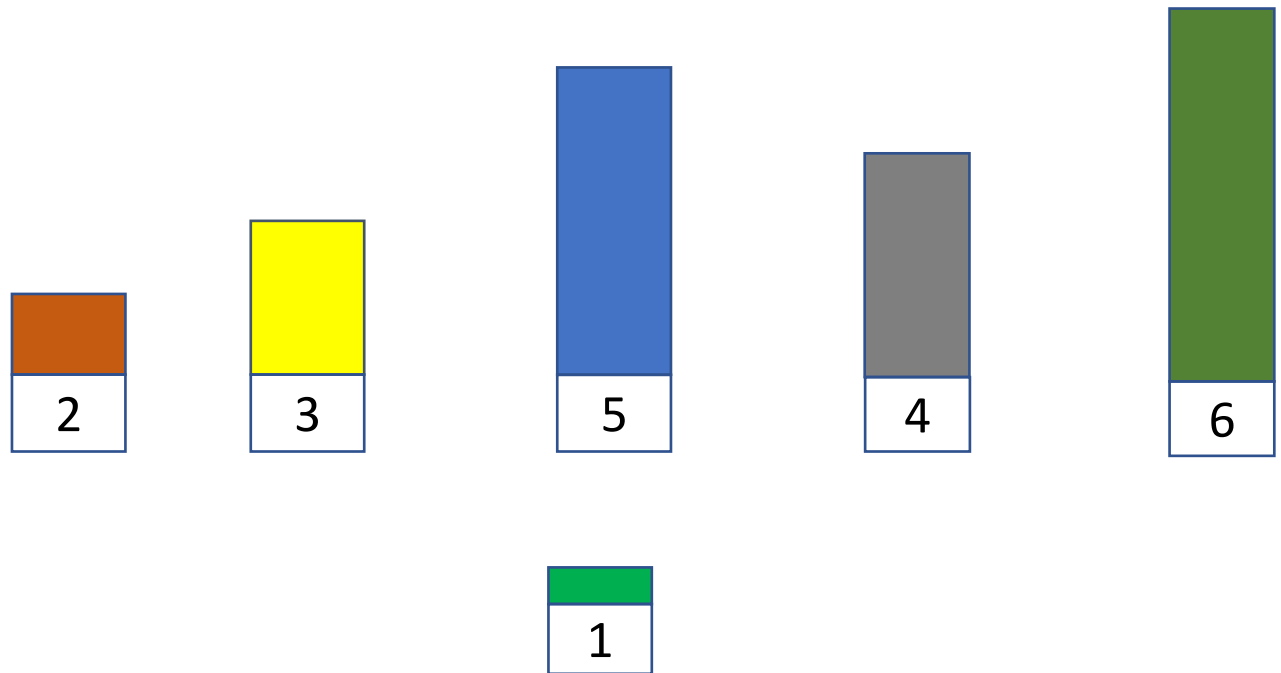
$i=4$ elemento a ordenar = 1



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

$i=4$ elemento a ordenar = 1

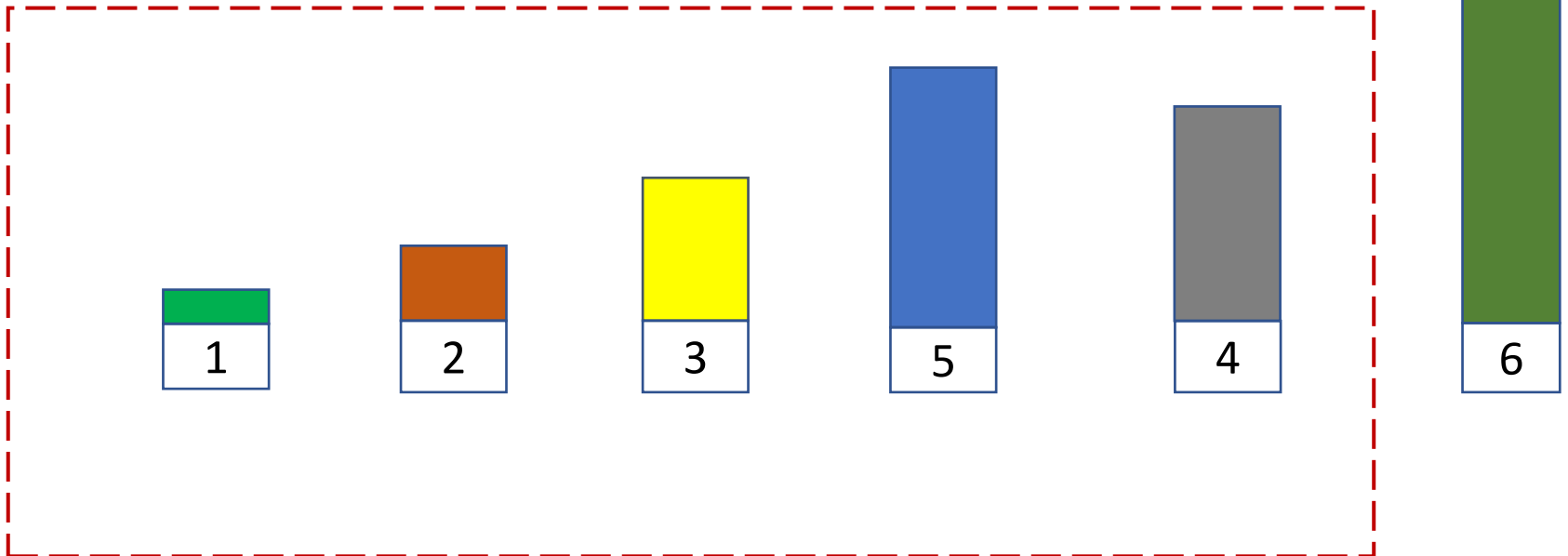


pos final= 1

2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

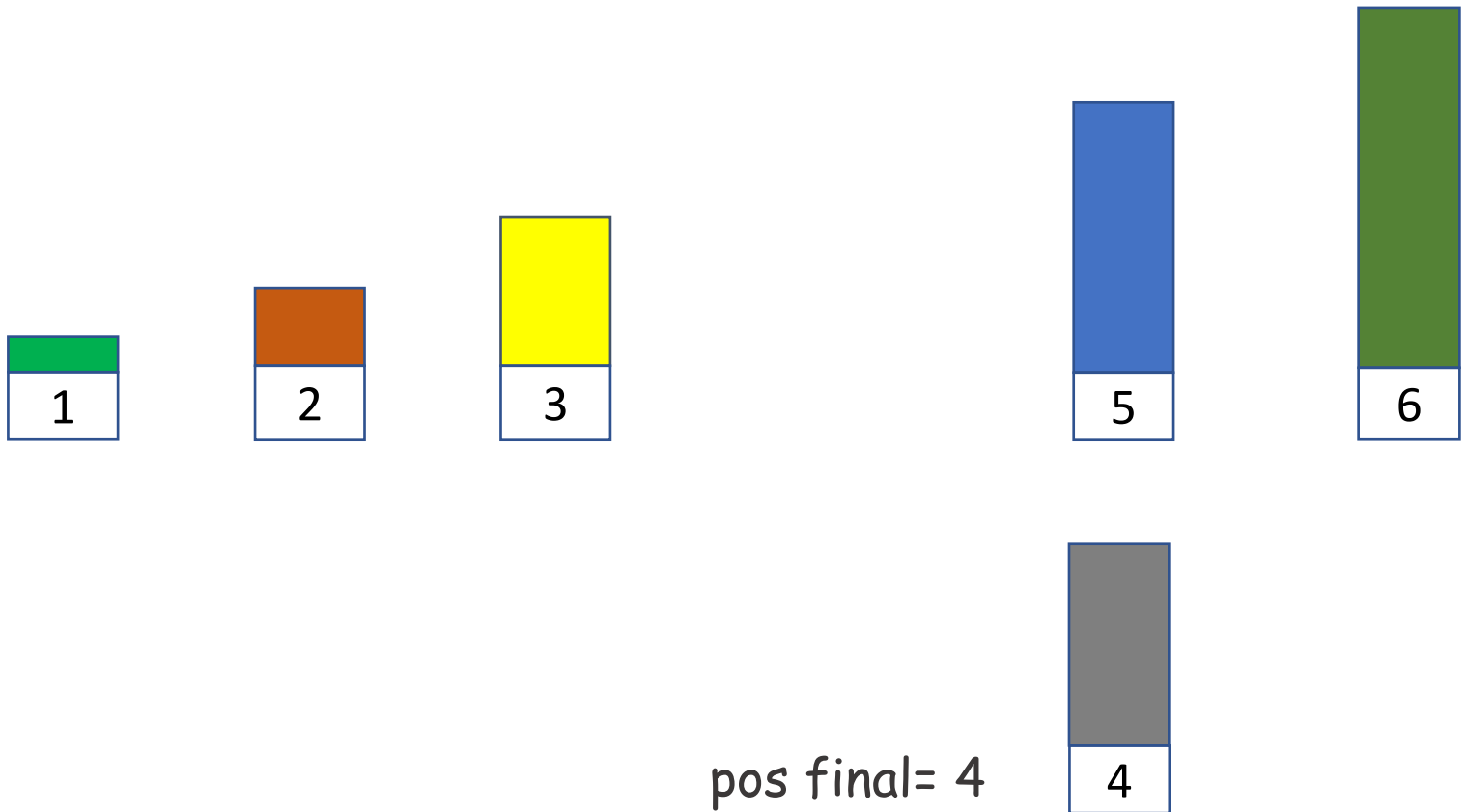
$i=5$ elemento a ordenar = 4



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

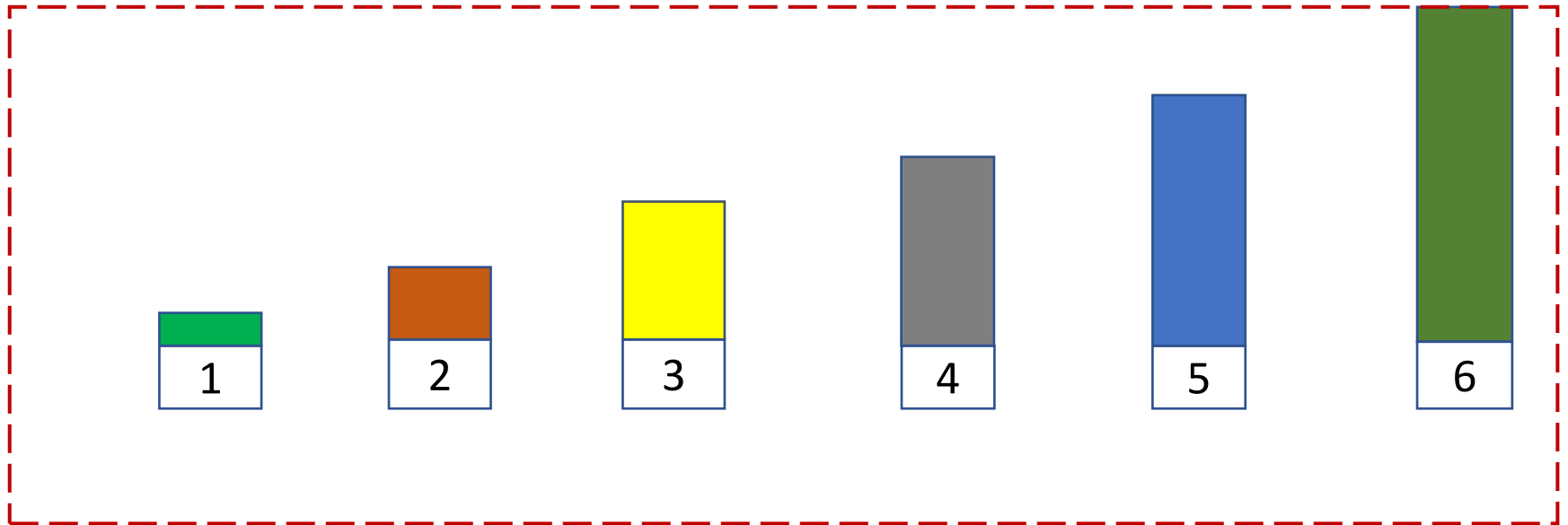
$i=5$ elemento a ordenar = 4



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

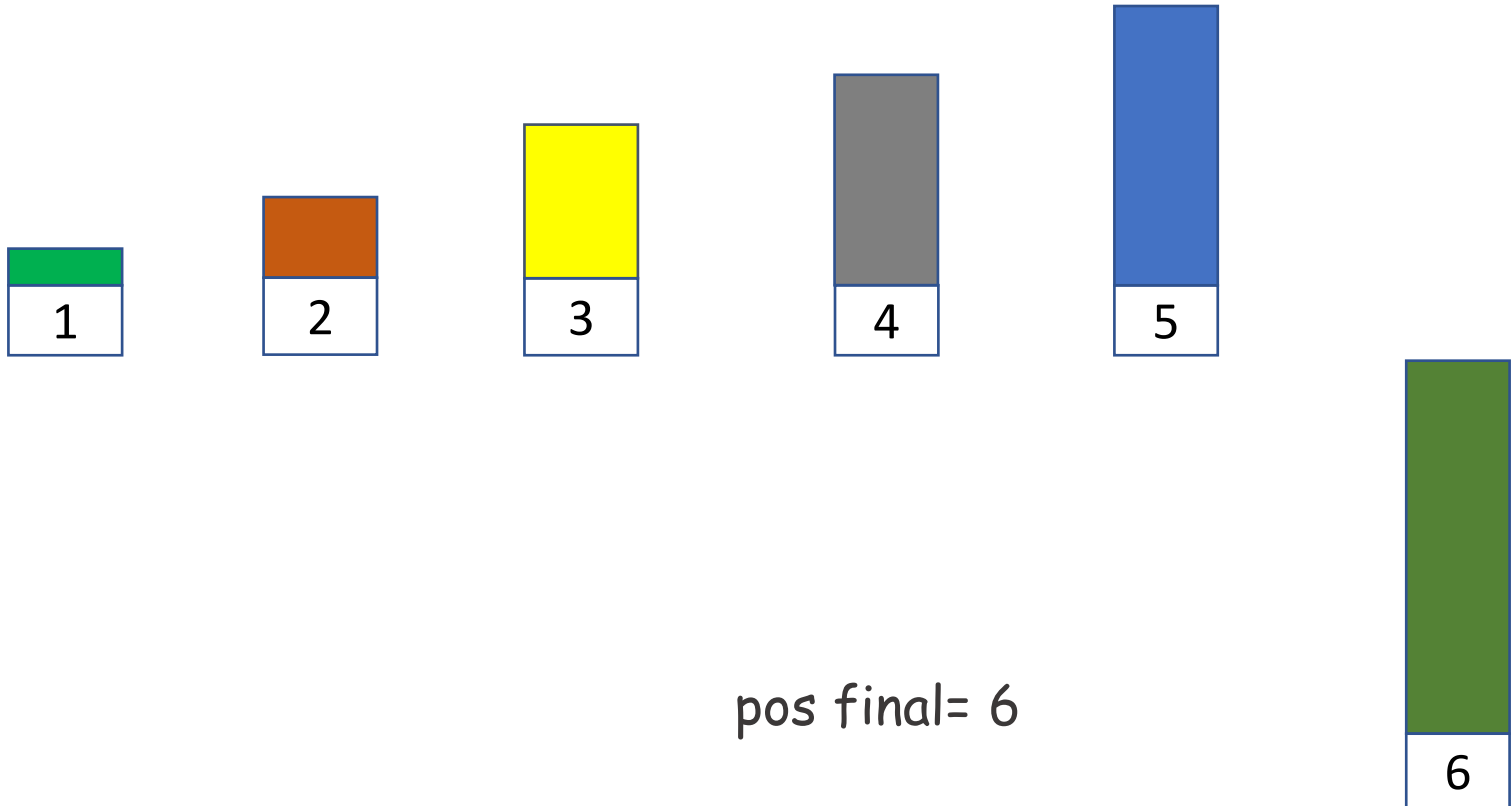
$i=6$ elemento a ordenar = 6



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

EJEMPLO: Ordenando de menor a mayor...

$i=6$ elemento a ordenar = 6



2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

2.1 PSEUCODIGO: Ordenar vector de n elementos de menor a mayor (n: dimensión lógica)

```
Repetir desde  $i:=2$  hasta  $n$   
     $actual := v[i]$       // Guardo elemento a ordenar  
     $j := i - 1$   
    Mientras ( $j > 0$ ) y ( $v[j] > actual$ )  
         $v[j+1] := v[j]$   
         $j := j - 1$   
     $v[j+1] := actual$   // Guardar elemento en  $v[j+1]$ 
```


2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

2.2 Análisis de casos

Link al simulador de ordenación por inserción

http://lwh.free.fr/pages/algo/tri/tri_insertion_es.html

```
Repetir desde i:=2 hasta n
    actual:= v[i]      // Guardo elemento a ordenar
    j:= i-1
    Mientras (j > 0) y (v[j] > actual)
        v[j+1]:= v[j]
        j:= j - 1
    v[j+1]:= actual    // Guardar elemento en v[j+1]
```

¿Qué pasa si los datos están ordenados de menor a mayor ?

1 | 2 | 3 | 4 | 5 | 6

¿Qué pasa si los datos están ordenados de mayor a menor ?

6 | 5 | 4 | 3 | 2 | 1

¿Qué modificaciones haría en este algoritmo si quisiera ordenar un vector de registros?

2. MÉTODO DE ORDENACIÓN POR INSERCIÓN

2.3 Análisis teórico

```
Repetir desde  $i:=2$  hasta  $n$   
   $\text{actual} := v[i]$       // Guardo elemento a ordenar  
   $j := i-1$   
  Mientras  $(j > 0)$  y  $(v[j] > \text{actual})$   
     $v[j+1] := v[j]$   
     $j := j - 1$   
   $v[j+1] := \text{actual}$   // Guardar elemento en  $v[j+1]$ 
```

C: Nro comparaciones (de elementos) y **M**: Nro de intercambios (de elementos)

Mejor caso

1 | 2 | 3 | 4 | 5 | 6

$$n-1 \leq C \leq n(n-1)/2$$

$$2(n-1) \leq M \leq 2(n-1) + n(n-1)/2$$

Peor caso

6 | 5 | 4 | 3 | 2 | 1



Ejercitación

ACTIVIDAD 1

a. En Geany, edite, compile y ejecute el siguiente programa

```
program NumAleatorio;  
  
var num: integer;  
  
begin  
    Randomize;  
    num := random (100); {valores en el intervalo 0 a 99}  
    writeln ('El numero aleatorio generado es: ', num);  
    readln;  
end.
```

b. ¿Qué hace el programa?



Ejercitación

ACTIVIDAD 2

Implemente un programa que contenga e invoque a los siguientes módulos:

- a. Módulo **CargarVector** que cargue un vector de enteros con valores aleatorios, hasta que llega el valor 0 o haber cargado 20 elementos como máximo.
- b. Módulo **ImprimirVector** que muestre los elementos del vector.
- c. Módulo **BuscarElementoVD** que busque un valor X sobre el vector desordenado y devuelva su posición o 0 en caso de no existir.
- d. Módulo **OrdenarPorInsercion** que ordene el vector de enteros cargado con anterioridad. Para ello utilice el *pseudocódigo* analizado anteriormente.
- e. Módulo **BuscarElementoVO** que busque de manera eficiente un valor X sobre el vector ordenado y devuelva su posición o 0 en caso de no existir. Para ello utilice Búsqueda Binaria (o dicotómica).



Ejercitación

ACTIVIDAD 3

Implementar un programa que procese la información de los jugadores de un torneo de básquet. De cada jugador se lee: dni, nombre y altura. El ingreso de los jugadores finaliza cuando se lee el dni 0 (que no debe procesarse). Se pide:

- Cargar la información en una lista. Los jugadores deben quedar en el mismo orden que fueron leídos.
- Mostrar la información almacenada en la lista.
- Generar un vector con aquellos jugadores que superan el promedio de altura. Se sabe de antemano que como máximo son 20.
- Mostrar la información almacenada en el vector.
- Ordenar el vector de jugadores por dni (ascendente)
- Mostrar el vector ordenado.
- Buscar en el vector al jugador cuyo dni coincide con uno ingresado e imprimir su altura.

NOTA: Reutilice los módulos implementados con anterioridad.